**University of the Punjab**

**Gujranwala Campus**

**Department of Information Technology**



**Assignment: Computer Vision**

**Prepared by:**

**Syed Hasnain Kazmi**

**Roll no:**

**BIT21057**

**Submitted to:**

**Miss Fouqia Zafeer**

**EXERCISE 1:**

**Write a program which can read an image as an input and do the following automatically. Show the results of all steps.1.**Find the type of image: binary, gray or RGB.2.Find the issue in image, over dark, over bright, low contrast, or normal. (Hint: can use histogram).3.Resolve the issue if any and show the final image after enhancement. 4.Test your program on following images

**CODE:**

function main()

% Test images (Replace these with actual file paths)

testImages = {'flower.jpeg', 'house.jpeg', 'nature.jpeg'};

% Process each image

for i = 1:length(testImages)

fprintf('Processing: %s\n', testImages{i});

processImage(testImages{i});

end

end

function processImage(filePath)

% Read the input image

image = imread(filePath);

% Step 1: Detect the type of image

imageType = detectImageType(image);

fprintf('Image Type: %s\n', imageType);

% Step 2: Analyze the image for issues

issue = analyzeImageIssues(image);

fprintf('Detected Issue: %s\n', issue);

% Step 3: Resolve issues and enhance the image

enhancedImage = resolveIssues(image, issue);

% Display the original and enhanced images

figure;

subplot(1, 2, 1);

imshow(image);

title('Original Image');

subplot(1, 2, 2);

imshow(enhancedImage);

title('Enhanced Image');

end

function imageType = detectImageType(image)

% Detect whether the image is Binary, Grayscale, or RGB

if ndims(image) == 2

uniqueValues = unique(image);

if length(uniqueValues) == 2

imageType = 'Binary';

else

imageType = 'Grayscale';

end

elseif ndims(image) == 3

imageType = 'RGB';

else

imageType = 'Unknown';

end

end

function issue = analyzeImageIssues(image)

% Analyze the image for over dark, over bright, low contrast, or normal

if ndims(image) == 3

grayImage = rgb2gray(image); % Convert RGB to Grayscale

else

grayImage = image;

end

% Compute mean and standard deviation of pixel intensities

meanIntensity = mean(grayImage(:));

stdIntensity = std(double(grayImage(:)));

% Classify the issue based on thresholds

if meanIntensity < 50

issue = 'Over Dark';

elseif meanIntensity > 200

issue = 'Over Bright';

elseif stdIntensity < 40

issue = 'Low Contrast';

else

issue = 'Normal';

end

end

function enhancedImage = resolveIssues(image, issue)

% Resolve the detected issue in the image

switch issue

case 'Over Dark'

enhancedImage = imadjust(image, [], [], 1.2); % Brighten

case 'Over Bright'

enhancedImage = imadjust(image, [], [], 0.8); % Darken

case 'Low Contrast'

if ndims(image) == 3 % RGB Image

labImage = rgb2lab(image);

L = labImage(:, :, 1);

L = histeq(L / 100) \* 100;

labImage(:, :, 1) = L;

enhancedImage = lab2rgb(labImage);

else % Grayscale Image

enhancedImage = histeq(image);

end

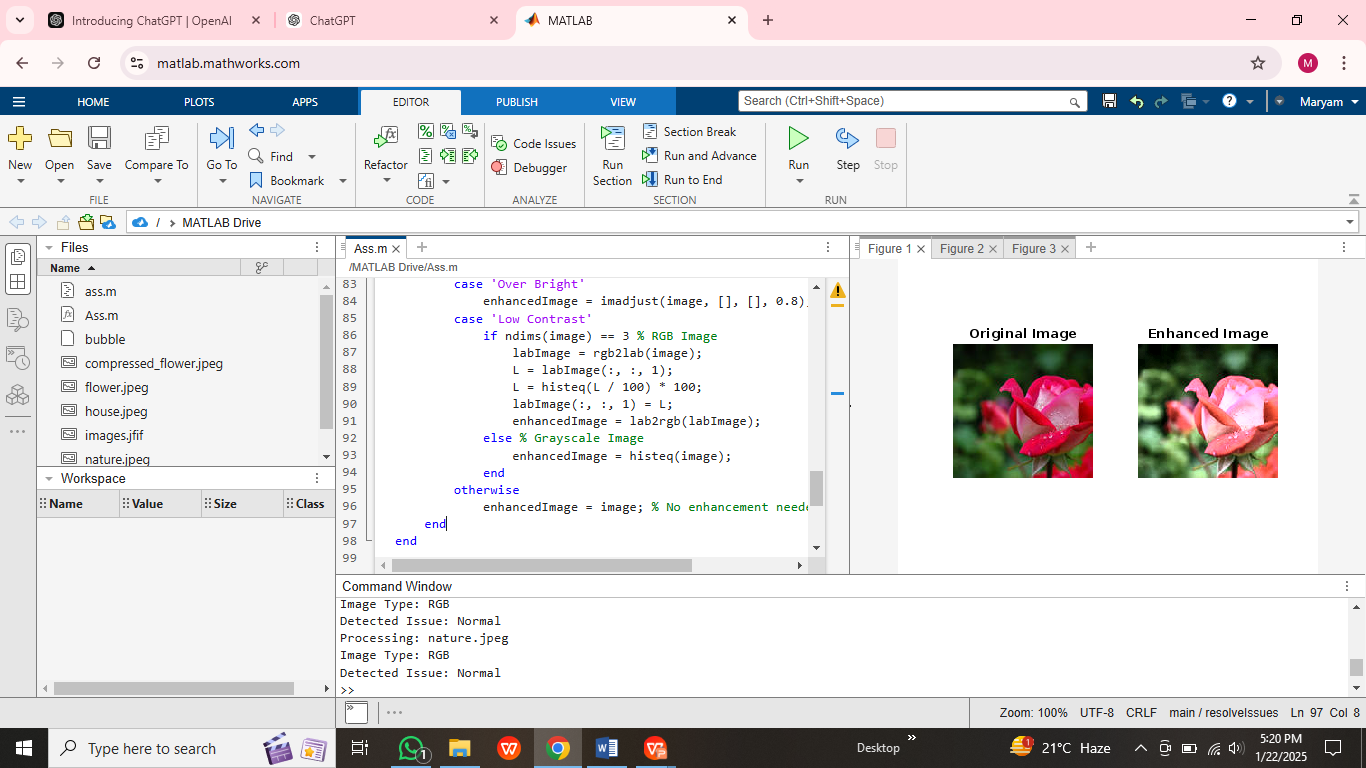
otherwise

enhancedImage = image; % No enhancement needed

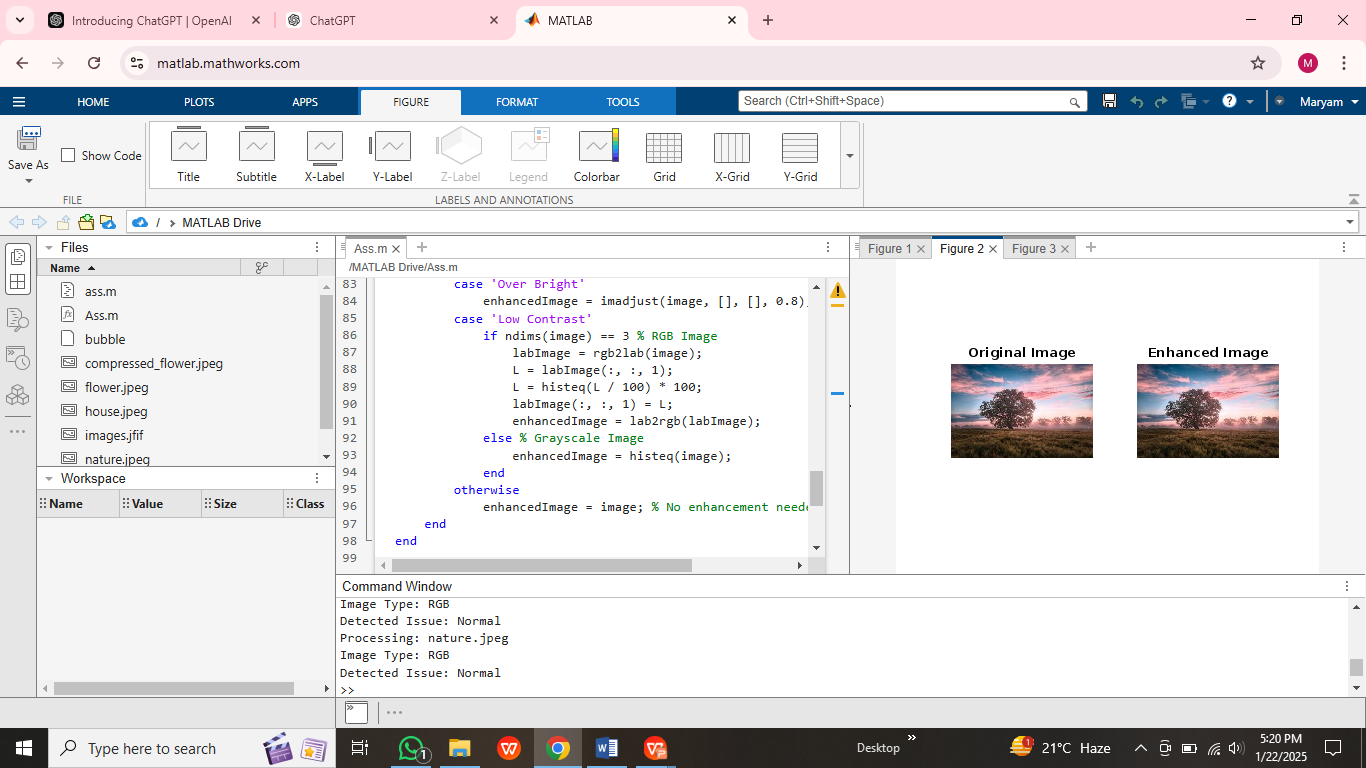
end

end

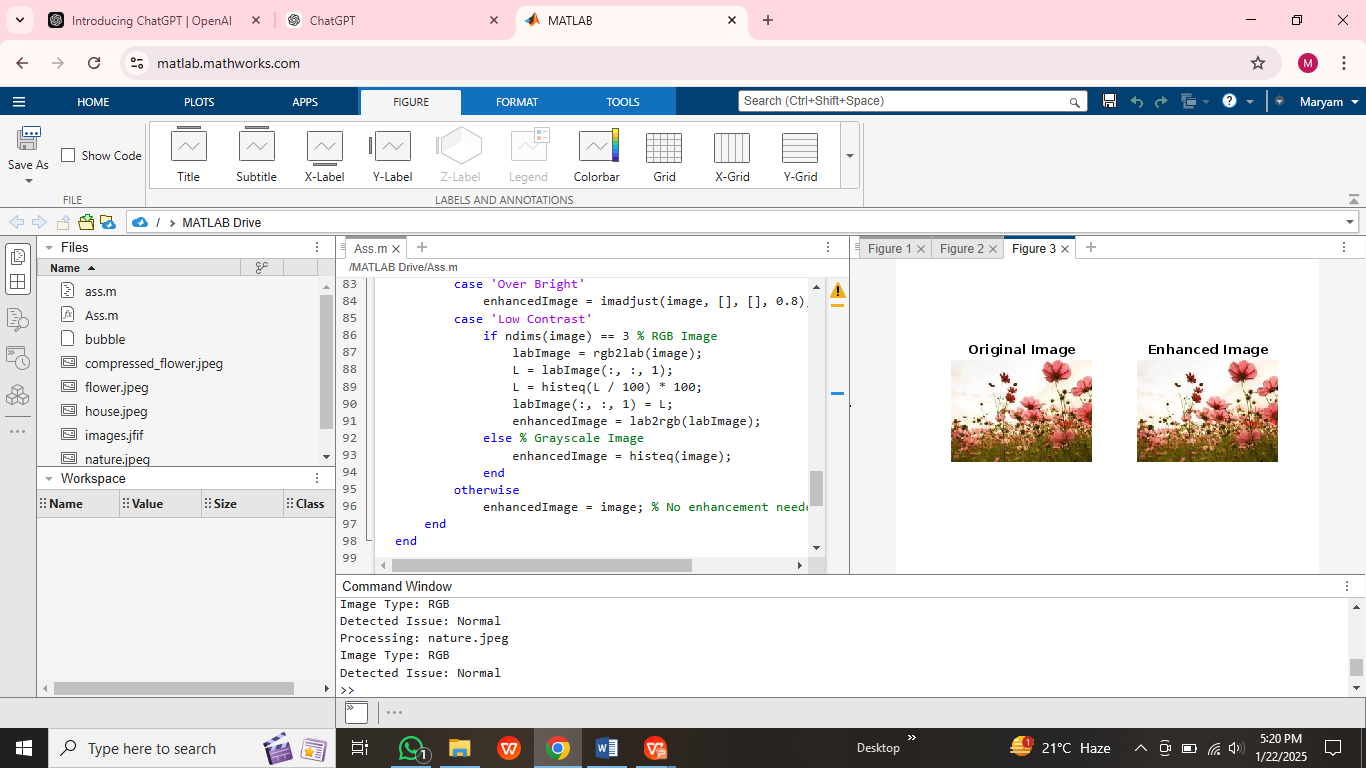
**OUTPUT 1:**

****

**OUTPUT 2;**



**OUTPUT 3;**



**EXERCISE 2:**

**Identify which intensity transformation was used on liftingbody.png to create each of the four results below. Write a script to reproduce the results using the intensity transformation functions.**

**CODE:**

function intensityTransformations()

% Read the original image

originalImage = imread('house.jpeg');

% Ensure the image is grayscale

if ndims(originalImage) == 3

originalImage = rgb2gray(originalImage);

end

% Apply transformations

% 1. Darkened image (Result 1)

result1 = imadjust(originalImage, [], [], 0.5); % Gamma correction with gamma < 1

% 2. Brightened image (Result 2)

result2 = imadjust(originalImage, [], [], 1.5); % Gamma correction with gamma > 1

% 3. High contrast image (Result 3)

result3 = histeq(originalImage); % Histogram equalization

% 4. Low contrast image (Result 4)

result4 = imadjust(originalImage, [0.3 0.7], [0.4 0.6]); % Adjust intensity range

% Display results

figure;

subplot(2, 3, 1);

imshow(originalImage);

title('Original Image');

subplot(2, 3, 2);

imshow(result1);

title('Result 1: Darkened');

subplot(2, 3, 3);

imshow(result2);

title('Result 2: Brightened');

subplot(2, 3, 4);

imshow(result3);

title('Result 3: High Contrast');

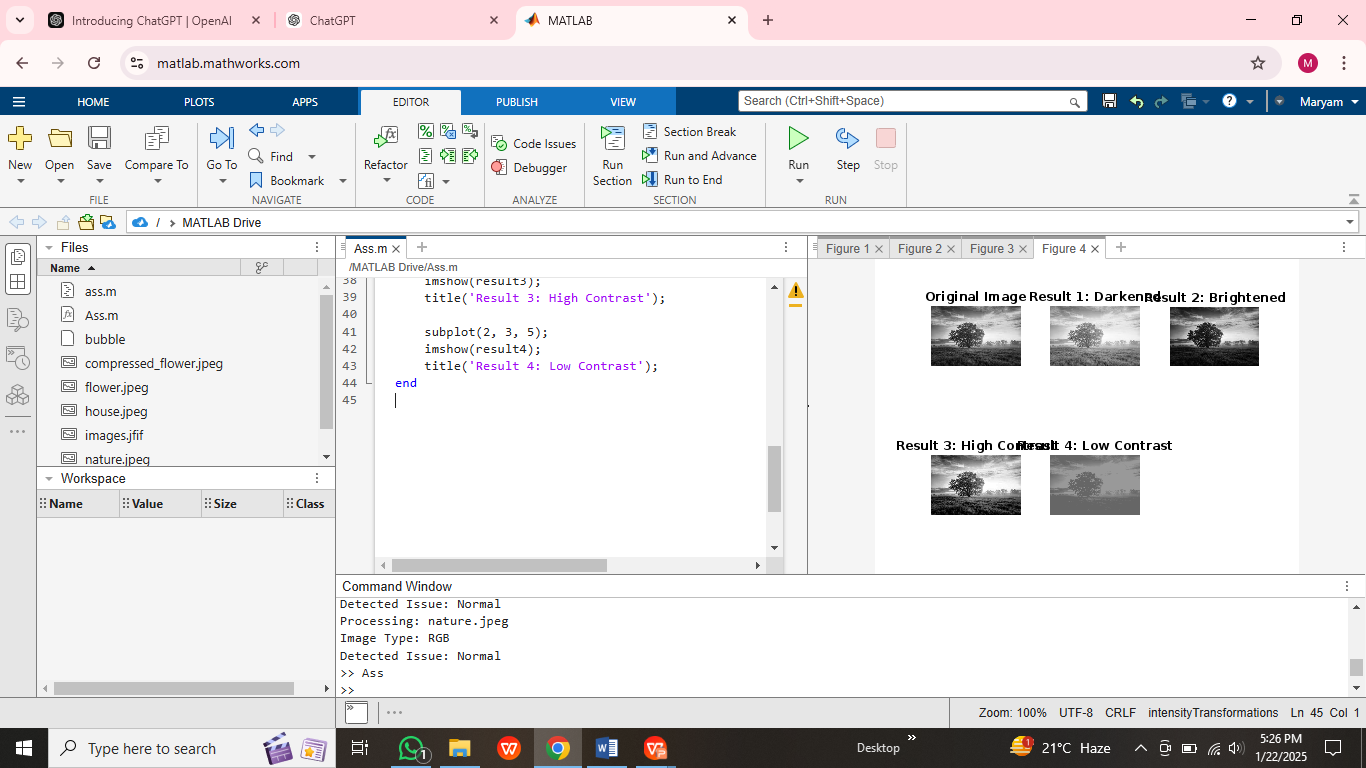
subplot(2, 3, 5);

imshow(result4);

title('Result 4: Low Contrast');

end

**OUTPUT:**

****